AMENDMENTS TO THE CLAIMS

1. (Currently Amended) A method in a computer system for scheduling tasks, the method comprising:

notifying a task executing on a parallel processor architecture having multiple simultaneously executing protection domains that the task is being preempted from utilizing the processor;

in response to the notification, receiving an indication from the task that it is ready to be swapped out and an indication as to whether the task is blocked;

swapping the task out; and

after swapping the task out,

when it is determined that the task can be swapped back in,

when it is determined that the task indicated that it was blocked,

when it is determined that an event occurred that may cause the task to unblock,

selecting a protection domain in which to execute the task,

swapping the task in, and

executing the task in the selected protection domain, and

when it is determined that an event did not occur that may cause the task to unblock, deferring the swapping in of the task until an event occurs that may cause the task to become unblocked, and

when it is determined that the task did not indicate that it was blocked,

selecting a protection domain in which to execute the task,

swapping the task in, and

executing the task in the selected protection domain

wherein the selected protection domain is different from the protection domain in which the task was originally executing so that when the task is swapped back in, the task executes within a different protection domain than the one in which it was originally executing.

2. (Original) The method of claim 1 wherein the computer system is a multithreaded computer system.

3. (Original) The method of claim 1 wherein in response to the notification, the task saves its state.

4. (Original) The method of claim 1 wherein the event is an indication from an operating system.

5. (Original) The method of claim 1 wherein the indication of whether the task is blocked includes an identification of a thread of the task that is blocked.

6. (Original) The method of claim 1 including tracking a number of threads of the task that are blocked.

7. (Currently Amended) A multi-processor computer system for scheduling tasks, each processor of the multi-processor computer system having multiple simultaneously executing protection domains, the system comprising:

a component that notifies a task that the task is being preempted from utilizing the processor;

a component that, in response to the notification, receives an indication from the task that it is ready to be swapped out and an indication as to whether the task is blocked;

a component that swaps the task out;

a component that determines that the swapped out task can be swapped back in;

a component that, when it is determined that the swapped out task can be swapped back in and an indication that the swapped out task is not blocked was not received, defers the swapping in of the swapped out task until an event occurs that may cause the swapped out task to become unblocked; and

a component that, when it is determined that the swapped out task can be swapped back in and an indication that the swapped out task is not blocked was received,

selects a protection domain in which to execute the swapped out task when it is swapped in, wherein the selected protection domain is different from the protection domain in which the task was originally executing,

swaps the swapped out task in, and

executes the task in the selected protection domain

so that when the task is swapped back in, the task executes within a different protection domain than the one in which it was originally executing.

8. (Original) The system of claim 7 wherein the computer system is a multithreaded computer system.

9. (Original) The system of claim 7 wherein in response to the notification, the task saves its state.

10. (Original) The system of claim 7 wherein the event is an indication from an operating system.

11. (Original) The system of claim 7 wherein the indication of whether the task is blocked includes an identification of a thread of the task that is blocked.

12. (Original) The system of claim 7 including a component that tracks a number of threads of the task that are blocked.

13. (Currently Amended) A computer-readable storage medium encoded with instructions for signaling deferred swapping in of a task executing on a parallel processor architecture having multiple simultaneously executing protection domains, by a method comprising:

when it is determined that a thread of the task is blocked, sending an indication to an operating system that the task is ready to be swapped out and that the thread is blocked;

after swapping out the task,

when it is determined that the task is ready to be swapped in,

sending an indication to the operating system that the task is ready to be swapped in;

when it is determined that the thread is no longer blocked,

selecting a protection domain in which to execute the task,

swapping the task in, and

executing the task in the selected protection domain wherein the selected protection domain is different from the protection domain in which the task was originally executing so that when the task is swapped back in, the task executes within a different protection domain than the one in which it was originally executing, and

when it is determined that the thread is blocked, sending an indication to the operating system that the thread is blocked, so that the operating system can defer swapping in the task

until an event occurs that may cause the thread to become unblocked.

14. (Canceled)

15. (Previously Presented) The computer-readable medium of claim 13 including incrementing a variable relating to a number of blocked threads wherein a task is blocked when all its threads are blocked.

16. (Previously Presented) The computer-readable medium of claim 13 including receiving an indication from the operating system that a thread is no longer blocked.

17. (Previously Presented) The computer-readable medium of claim 16 including decrementing a variable relating to a number of blocked threads.

18. (Previously Presented) The computer-readable medium of claim 13 wherein the task does not know which of the multiple protection domains on which it is executing.

19. (Previously Presented) A method in a computer system for scheduling tasks, the method comprising:

notifying a task executing on a parallel processor architecture having multiple simultaneously executing protection domains that the task is being preempted from utilizing the processor;

in response to the notification, receiving an indication from the task that it is ready to be swapped out and an indication as to whether the task is blocked that includes an identification of a thread of the task that is blocked;

determining that the task can be swapped back in;

when an indication that the task is blocked is received, deferring the swapping in of the task until an event occurs that may cause the task to become unblocked; and

when the task is swapped back in, executing the task within a different protection domain than the one on which it was originally executing.

20. (Previously Presented) The method of claim 19 wherein the processor has 16 simultaneously active protection domains.

21. (Previously Presented) A method in a computer system for scheduling tasks with multiple streams, the method comprising:

notifying a task that the task is being preempted from utilizing the processor;

in response to the notification, receiving an indication from the task that it is ready to be swapped out and an indication as to whether the task is blocked, the task being blocked as a result of all streams assigned to the task executing virtual processor code;

determining whether the task can be swapped back in; and

when it is determined that the task can be swapped in and an indication that the task is blocked was received, deferring the swapping in of the task until an event occurs that may cause the task to become unblocked.

22. (Previously Presented) The method of claim 21 wherein the task is not considered blocked when the virtual processor code for a stream is in the process of starting a thread.

23. (Currently Amended) A method in a computer system with a parallel processor architecture having multiple simultaneously executing protection domains with multiple streams for scheduling tasks with multiple threads, the method comprising:

notifying a task executing on the computer system that the task is being preempted from execution;

in response to the notification, receiving an indication from the task that it is ready to be swapped out and an indication as to whether the task is blocked, the task being blocked when all streams executing threads of the task are blocked;

determining whether the task can be swapped back in;

when it is determined that the task can be swapped in and an indication that the task is blocked was not received,

selecting a protection domain to swap the task into, wherein the selected protection domain is different from the protection domain in which the task was originally executing, and

swapping the task into the selected protection domain

so that when the task is swapped back in, the task executes within a different protection domain than the one in which it was originally executing; and

when it is determined that the task can be swapped in and an indication that the task is blocked was received, deferring the swapping in of the task until an event occurs that may cause the task to become unblocked.

24. (Currently Amended) A method in a computer system with a multi-processor architecture having multiple simultaneously executing protection domains with multiple streams for scheduling tasks with multiple threads, the method comprising:

receiving from a task executing on the computer system an indication that the task is blocked; and

in response to receiving the notification,

determining whether to swap the task out from use of the processors;

when it is determined that the task is to be swapped out from use of the processors,

notifying the task that it is to be swapped out;

in response to the notifying, receiving from the task an indication that the task is ready to be swapped out;

after receiving the indication that the task is ready to be swapped out, swapping out the task;

after the task is swapped out,

determining whether an event occurs that may cause the task to become unblocked;

determining whether the task can be swapped back in; and

when it is determined that the task can be swapped in and it is determined that an event occurred that may cause the task to become unblocked,

selecting a protection domain in which to execute the task wherein the selected protection domain is different from the protection domain in which the task was originally executing so that when the task is swapped back in, the task executes within a different protection domain than the one in which it was originally executing,

swapping the task in, and

executing the task in the selected protection domain, and

when it is determined that the task can be swapped in and it is determined that an event did not occur that may cause the task to become unblocked, deferring the swapping in of the task until an event occurs that may cause the task to become unblocked.

25. (Previously Presented) The method of claim 24 wherein the task is blocked when all threads of the task are blocked.

26. (Currently Amended) ~~The method of claim 24~~A method in a computer system with a multi-processor architecture having multiple simultaneously executing protection domains with multiple streams for scheduling tasks with multiple threads, the method comprising:

receiving from a task executing on the computer system an indication that the task is blocked; and

in response to receiving the notification,

determining whether to swap the task out from use of the processors;

when it is determined that the task is to be swapped out from use of the processors,

notifying the task that it is to be swapped out;

in response to the notifying, receiving from the task an indication that the task is ready to be swapped out;

after receiving the indication that the task is ready to be swapped out, swapping out the task;

after the task is swapped out,

determining whether an event occurs that may cause the task to become unblocked;

determining whether the task can be swapped back in; and

when it is determined that the task can be swapped in and it is determined that an event occurred that may cause the task to become unblocked,

selecting a protection domain in which to execute the task,

swapping the task in, and

executing the task in the selected protection domain, and

when it is determined that the task can be swapped in and it is determined that an event did not occur that may cause the task to become unblocked, deferring the swapping in of the task until an event occurs that may cause the task to become unblocked

wherein the task is blocked when all streams assigned to the task are executing virtual processor code.

27. (Previously Presented) The method of claim 26 wherein the task is not blocked when a stream is executing virtual processor code to start a thread of the task.